University of California at Berkeley
Mechanical Engineering

ME 39E - Animating Physics

Fall, 1999

2 units, graded

Instructor: David M. Auslander
5120 Etcheverry Hall
510-642-4930
dma@me.berkeley.edu

GSI: Pam Sirivedhin <pam@me.berkeley.edu>

Course Objective

Produce physically realistic, computer-based animations. In the course of doing this, use the laws of basic physics to produce mathematical models of the objects in the artistic concept. Then, convert the mathematical models to numerical and finally computational form to bring the animation to life.

Why do this? Animations are fun and have an important place in our personal and professional lives. Incidentally, doing this illustrates the unity of the subject matter in physics, math and computing. An animation is all of these at the same time.

Course Operation

The class meets once a week (Tuesdays, 3-5, 2109 Etcheverry). The purpose of the class is to discuss and try out new techniques and methods in the creation of animations. These could be related to computing, physics, or math, depending on each student's current problems.

As much as possible, the class proceeds by solving complete problems right from the start, first relatively simple problems, then more complex problems. Move through levels of complexity as suits your background and temperament.

The classes will consist of four parts:

> **Questions**: the most important part! What's causing problems, doesn't work right, doesn't make sense, etc.
>
> **Student Presentation:** Short presentations on recently completed or current projects. Presentations will be five minutes (timed!) and will be computer based (Power Point).

Discussion of New Material: More detailed discussion of any topics that have not been adequately addressed in the question period.

Try Stuff and Consultation: Individual work with instructor, TA, and other students.

Class Notes and Texts

Notes for the class are available from:

- On the class web site (http://euler.Berkeley.EDU/~dma/res.html – go to the ME39E section). Both the viewable HTML file and the downloadable PDF file are available. PDF files are in Adobe Acrobat format. You can get a free viewer for these files from http//www.adobe.com.
- On paper, from COPY WORLD, 2154 UNIVERSITY AVE, BERKELEY - 849-9701 (between Shattuck and Oxford).

Other texts are for the physics, math, and computing. Use whatever texts you have for physics and math. It may take some reading between the lines to put the material into the format we use for animations! Calculus is **not** required. On the other hand, if you have taken calculus don't waste the experience! It will add considerably to your understanding of what we are doing. On the other hand, trigonometry and algebra will be used very heavily.

Matlab, software from Mathworks, Inc., is the basic platform for all of the computing in this class. It is one of several widely used packages in engineering, both in universities and in industry. Computers with Matlab installed are available to you in the class lab. You can use these any time the room is not being used for another class (there are also computers you can use in adjoining rooms. You can get a card key for access from Joey Cruz in room 2115 Etch – there is a charge for the key and a deposit as well. Check his hours for key transactions.

Matlab is also available for home use, which is highly recommended if you have a computer appropriate to run it (any reasonably modern Windows-PC or Mac should work). There are two products for students (they are restricted for sale to students only): Matlab Student Edition v5.3 and Matlab Student Version v5.3. The first is distributed by Prentice-Hall (a book publisher) and the second is distributed by The Mathworks directly. The Student Edition puts some limits on data size but has more built-in components than the Student Version which doesn't limit data size. Either will work for this class. These are both new versions of this product so we are checking actual availablility. The price is around $100.

The student editions include an extensive user's manual for Matlab but it isn't written in tutorial form. The recommended text, *Mastering Matlab 5*, Hanselman & Littlefield, Prentice-Hall, ISBN#0-13-858366-8, has more of a new-user perspective. (If you don't find the Student Edition in the bookstore under our course number, try looking under Engineering 77.)

Assignments and Grading

1. Weekly reports: Written, **no more than one page of text plus an optional page of graphics**. The report should outline what you have done in the past week. For the first 8 or 10 weeks this will be a sequence of exercises. At the end of the semester the weekly reports will document progress on your final project. Each weekly report is worth 3 points. 36 points maximum credit for progress reports (*i.e.* you can miss a couple). Can be submitted by email (to Pam). No credit for late progress reports (but see 36 point maximum credit). Weekly reports are due at the beginning of class each week.

2. Presentations: Three presentations, roughly one every four weeks (see above for details on these). 5 points each.

3. A final report, written, no more than 4 pages. This report should describe your final project. It is due at the last class meeting for this class (Tuesday, Nov 30). Can be submitted by email (to Pam). 10 points.

A note on page limits: for reports listed with page limits, only the number of pages up to the limit will be read! A page should have normal margins and a type size no smaller than 12 pt. Figures can be made fairly small, but must be readable.

Cooperative Work

All sorts of interaction and cooperation are encouraged. Talk to other people, work together on concepts, consult on implementation issues, etc. However, remember that in the end *you are responsible for what you produce*. Give credit to others where it is due and take credit for incorporating as many good ideas from as many places as possible into your creation.

Curriculum

This is a seminar. What that means is that this course is not prerequisite to any other course so that what is accomplished here is very open. It also means that individual students do not have to accomplish the same things. The only constant is the theme, animating physics. Beyond that, every student is free to his/her interpretation and to proceed according to that interpretation.

With that in mind, many of you might like some guidance on how to proceed, particularly at the beginning. The following are thus suggestions so take them as such and feel free to improvise. There is a set of suggested exercises that you can use to put these guidelines into practice. You are free to use them or substitute others you find more interesting or challenging.

How much time to spend on this course: the general interpretation of workload is that 1 unit represents about 4 hours per week of work, including class time. This course should thus be about 8 hours per week of total time commitment.

Week #1: Orientation and setup. By the end of the week you should be able to run the sample problems and change parameter values in the samples to vary their behavior. If you are setting up the student edition of Matlab on your own computer get that done.

Read this document thoroughly!

Weeks #2-4: Learn Matlab. Do it from a book, attend extra class sessions, learn it from or with a friend. How to do this? Learning a language is a largely personal matter The suggestions that follow might suit your style, but, if they don't, don't worry. You will work out a procedure that works!

Learning a language entails a certain amount of frustration -- be it a natural, spoken language or a computer language. In both cases the memorization of vocabulary and rules of syntax are part of that frustration. In the case of natural language, though, a lot of the frustration comes from ambiguity -- the specific meaning of a phrase, or even a word, may not be obvious without examining the larger context. In computer languages, it is often the preciseness that is frustrating -- the result will be *exactly* as written, which may well be far from what you intended.

Matlab is an interactive language so take advantage of that. Because it is interpretive, it can give you immediate feedback. Never do anything that takes more than 5 minutes without trying it out. Start by experimenting with the command line. Command line computing used to be much more common than it is, so using a command line might not be very familiar. The "history" feature and the ability to use cut-and-paste with command line entries can be very powerful.

Make sure you can make lots of the direct commands work. Experiment with arrays and matrices (the "mat" in Matlab stands for matrix). Go on to the functions. There are untold numbers of these so pick a few from each category. Make sure to include the *plot()* function in your experiments.

Matlab is a general-purpose data manipulation and graphics language. You should be able to make use of it in many other numerically-oriented classes.

Next, start with the "flow control" statements, *if*, *for*, *while*, and *switch*. These can be used in simple constructs from the command line. When that becomes too constraining, switch to using separate *script* files. These are files that contain exactly the lines you could have typed at the command line to achieve a specific result. The difference is that script files can get much more complicated than anything you can handle conveniently at the command line.

From script files it is only a small step to functions. These look a lot like script files, but add the use of *arguments* and *return values*. The advantage to functions is that the arguments and return values give them a degree of isolation that can be exploited to make programs easier to organize.

All through this, keep going back to the animation samples. For everything you learn figure out a way to use it to make a small (very small) change in one of the samples. Put the change in and try it out.

Enough for learning languages in abstraction! Now it is time to put this knowledge to use. Yes, it is true that you will constantly run up against things you don't quite know how to do. By this time, however, with the motivation of needing to solve a problem, you won't have too much trouble figuring out how to do it.

Weeks #5-7: Physics, math and computing. Read the class notes carefully and review your physics and math. Take homework problems from the physics text and pose them in the format given in the class notes. This format will be suitable for numerical rather than analytical solution. Using your knowledge of Matlab, do some simple script files that produce numerical solutions of the homework problems and compare them to the answers in the text or that you generate by the methods given in the text. Use the *plot()* facilities of Matlab to get graphs of the answers.

The numerical approach to physics that we use here is very different from the approach you have seen in previous physics classes. Ask lots of questions to cement your understanding.

The work in physics should be done in parallel with continued work on computing and animation…

Change your approach to computing. Instead of doing things in order to learn some specific feature or technique, start with an animation concept and make it happen. You can start this by modifying some of the sample problems. Pick a problem and think of what you would like it to do that it doesn't do now (add a new element, change the behavior, change the geometry, change the animation concept for the same physics, etc.) After some success with this approach, design a simple animation from scratch. Use one of the sample programs as a template for starting your animation. Pick the sample that is closest even if it only bears a passing similarity. It will still give you the security of having a working starting point.

Weeks #8-10: Conceive of, plan and implement an animation of a bit more scope. As with any creative endeavor, the key to success is doing lots of studies. A study is a design and implementation of only part of the intended creation. Every museum is full of studies that artists have done before tackling major concepts in paintings or sculpture. Typically, studies are used to try out things that you are not certain of. The more creativity, the more uncertainty! Try not to be too grandiose in your intentions. Keep in mind the limits of time you have to spend on completion.

Weeks #10->: You're on your own. Enjoy!

Suggested Exercises

The exercises below represent our guess at the minimum progress that me39e students should be able to accomplish. They are specific suggestions to implement the general progress outlined above. They are only suggestions; they are not set in "stone." If you find you can go faster or have other exercises you would prefer to do, please go ahead. Just make sure that the substitute exercises cover similar material. If you find that you cannot keep up with this pace, let us know as soon as possible. *Remember this is a seminar--we can tailor the material to suit individual needs!*

First Week: Run at least 5 of the sample animations. Do this either at home if you have access to Matlab or to the Matlab Student Edition/Version or in our lab (2109 Etch.) Experiment with changing various parameters. Specifically, write about what happens as DeltaT is made larger in TwoBilliardBalls_2.m and when one of the masses is made very small in MultipleSpringMass.m. Do the results make sense? What do you think caused these results?

Subscribe to the class mailing list (see the Communication section, below).

Just for this first week, the report should also include email address, computer/programming/Matlab experience, physics experience, time available for the Matlab session

Second week: Use help and lookfor to find out how to use basic commands such as pwd, cd, dir, if else, elseif, end, while, for, switch, plot, sin, cos. Find out what [], (), up&down arrows, %, :, &, |, ==, ~= do. (hint: type 'help ops')

Use these operators to write some simple programs. Here are some suggestions for short programs that use these operators:

Sum the integers from 1 to 1374.
Sum all of the integers divisible by 3 from 87 to 935.
Find the product of the integers from 1 to 12.
Find factorial(12) (see above!).
Find factorial(n).
Test to see whether a number is an integer (see ceil(), floor(), fix()).
Test to see whether one number is evenly divisible by another.
Test to see whether a number is prime.
Plot sin(x) from x = 0 to 10 radians.
Plot cos(x) from x = 90 to 360 degrees.
Find the hypotenuse of a right triangle given the sides.
Find the intersection of two lines (each line is specified by two points).

You'll be quizzed about this in class!

Third week:
Part 1: Write simple Matlab programs simulating dropping a 1kg ball from a height of 5 m. Calculate — and print and plot on the screen -- the position of the ball at 0.1 seconds

intervals until impact using a) traditional physics equation. b) the difference equation. Are the results the same? Why or why not?

Part 2: Consider the same system as the above but now with air resistance proportional to the velocity of the ball. (i.e., F = b.v) Can you calculate the position of the ball using traditional physics equations? Modify the program from part 1b so that it takes into account the air resistance.

Fourth week:
Part 1: Make modifications to at least two of the sample animations. The changes should affect the animation rather than just the initial appearance. Some suggestions for changes:

Change size, orientation, or color as the object moves. Base the new data on position (note: there are a couple of samples that show how to do this). Note that the **type** of object cannot be changed during animation – that is, the object cannot be changed from a rectangle to a circle or a patch to a rectangle, etc..

Change the billiard ball example so that there is some loss of velocity with each collision.

Change the simple pendulum example so that the pendulum bob (which is a square) behaves as if it were on a pivot instead of rotating as it oscillates. This would cause the sides of the square to remain horizontal and vertical as it oscillates. Hint: this will require that the graphic representation of the pendulum be changed from a single "patch" to two rectangles.

Make the billiard balls “flash” (ie, change color) as they collide.

Part 2: Expand the result from part 1b the previous week’s into a 2-D problem. Instead of just dropping the ball, you are now throwing it horizontally at 5m/s. You'll also have two results to present: the x and the y positions.

Part 3. Animate the results of part 2.

Part 4: Now add the air resistance like in part 2 of the previous assignment. Remember that the direction of the air resistance is opposite the direction of the velocity.

Exercises for subsequent weeks will be distribute at a later date.

Presentations

In-class presentations are done using Power Point, Microsoft software that is used to construct presentations. There is a projector in the classroom that will take the image directly from the computer. In order to avoid long setup delays, any animations you want to include in the presentation must first be converted to video files. This is done with another program called SnagIt. It can capture static images, text or video from the computer’s screen. We have a brief tutorial below on using it for video capture. You can get more information from its “help” section or from the manufacturer’s web site,

www.techsmith.com. You can also get information on purchasing it for home use if you want to (it is a shareware program).

Using SnagIt to Turn Matlab Animations into Videos

Start Matlab and run animatewin
Start SnagIt
Click on "Video Capture"
From menu
- Input→Window
- Output→File and Preview Window
- Output→ properties → Video File
  - Uncheck Auto Configure AVI Options
  - Check Enable Slow AVI Capture Mode
  - Click Frame rate Setup
    - Start with 2 frames/second capture and 20 frames/second playback. (You'll have to experiment with this setting to get a nice result. See also the Troubleshooting section below for more information.)

Click on the animation window and hit control-shift-P to start the capture
A red window will appear around the area you will capture. Picking the title bar ensures that the whole window is picked.
Hit "start animation" button in the animation window.
Hit "Start" in the SnagIt dialog box
Capturing now in progress. Don't work on anything else while waiting.
Hit control-shift-P again to end capture. Pick the "stop" button if all went well….or "cancel" if you'd like to start over.

This will bring up the preview window.
To preview the video, pick the button with the smaller black right arrow or pick View→play

If you like what you see and would like to save it, pick the green arrow (or hit enter, or File→ Finish output) and save your file. If not, pick the red square (or hit escape, or File → Cancel output) to cancel the whole thing and start over.

Note: You may need to adjust the speed of the animation to get the desired result. For example, slowing down the animation (i.e., decrease DeltaT) often gives cleaner, smoother animation at the expense of speed. However, you must be careful in increasing DeltaT to speed up the animation since having a very large DeltaT can result in all sorts of errors and inaccuracies.

Troubleshooting:
Animation is very jerky: decrease DeltaT, or increase the frame capture rate.
Resulting video is slow: increase DeltaT, or increase the frame playback rate.
Resulting video is too fast: decrease DeltaT, or decrease the frame playback rate.
Visible animation errors: decrease DeltaT, debug your animation program.

Takes too long to capture even a short video: decrease the frame capture and playback rate

Remember these settings are computer dependent. On a faster computer, an animation with the same DeltaT will move faster. It also enables you to capture at a higher rate.

Communication

Official communications (due dates, changes in assignments, etc.) will be done in email and in class. You can get connected to the class email list by sending a message to majordomo@newton.berkeley.edu. The procedure is:

1. Address an email to majordomo@newton.berkeley.edu. Make sure this is done from the email account at which you want to receive class email.
2. Put the following lines in the *message* section of the email (leave the *subject* blank):
   subscribe me39e-class
   end
3. You should get a return email confirming your subscription.
4. If you want to get off the mailing list, use "unsubscribe" instead of "subscribe."
5. To find out all of majordomo's commands, send
   help
   end

You can send email to the whole class by addressing it to *me39e-class@me.berkeley.edu*.

If you do not have an email account you can check the following web page for information on how to get a free email account on UC's system:
http://bilink.berkeley.edu:8000/net.html.